```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler, StandardScaler, LabelEncoder, OneHotEncoder



# Load the dataset
df = pd.read_csv("/content/sample_dataset_with_nulls.csv")


# 1.Get the number of rows, columns, datatype, and summary stats of each column of a dataframe. Als
print("Number of Rows:", df.shape[0])
print("Number of Columns:", df.shape[1])
```

```
Number of Rows: 100
Number of Columns: 10
```

```
print("Data Types:\n", df.dtypes)
print("Summary Stats:\n", df.describe())
```

```
Data Types:
 ID                    int64
Name                 object
Age                 float64
Salary              float64
Department           object
Rating              float64
Experience            int64
Join_Date            object
Has_Certification      bool
Performance_Score     int64
dtype: object
Summary Stats:
                ID        Age        Salary      Rating  Experience  \
count  100.000000  90.000000     90.000000  100.000000  100.000000
mean    50.500000  37.588889  66913.388889    2.916000   20.470000
std     29.011492  12.234501  20883.453851    1.214119   10.907744
min      1.000000  18.000000  30206.000000    1.000000    0.000000
25%     25.750000  26.250000  51928.250000    1.775000   11.000000
50%     50.500000  38.000000  68988.000000    2.850000   21.000000
75%     75.250000  46.000000  82927.250000    3.900000   30.000000
max    100.000000  59.000000  99163.000000    4.900000   39.000000

       Performance_Score
count         100.000000
mean           73.780000
std            14.950815
min            50.000000
25%            61.750000
50%            75.000000
75%            85.250000
max            99.000000
```

```
print("Array Equivalent:\n", df.values)
```

```
 [52 'Person_52' 55.0 64766.0 'IT' 2.8 18 '2014-04-30' True 66]
 [53 'Person_53' 57.0 91087.0 'IT' 2.0 11 '2014-05-31' True 54]
 [54 'Person_54' 21.0 98840.0 'Finance' 1.4 8 '2014-06-30' True 80]
 [55 'Person_55' 19.0 84384.0 'Sales' 1.7 6 '2014-07-31' False 54]
 [56 'Person_56' 23.0 81005.0 'HR' 4.7 27 '2014-08-31' False 87]
 [57 'Person_57' 59.0 76576.0 'Marketing' 3.6 13 '2014-09-30' True 52]
 [58 'Person_58' 21.0 69353.0 'HR' 3.1 30 '2014-10-31' True 72]
 [59 'Person_59' 46.0 92003.0 'Sales' 3.6 18 '2014-11-30' False 86]
 [60 'Person_60' 35.0 82733.0 'Marketing' 2.7 15 '2014-12-31' True 86]
 [61 'Person_61' 43.0 95318.0 'Sales' 3.9 4 '2015-01-31' False 59]
 [62 'Person_62' 51.0 53664.0 'IT' 1.2 34 '2015-02-28' True 59]
 [63 'Person_63' 27.0 97172.0 'Marketing' 3.3 11 '2015-03-31' True 68]
 [64 'Person_64' 53.0 56736.0 'IT' 1.6 24 '2015-04-30' False 66]
 [65 'Person_65' 31.0 30854.0 'HR' 1.5 20 '2015-05-31' True 70]
 [66 'Person_66' 48.0 68623.0 'HR' 2.4 35 '2015-06-30' False 63]
 [67 'Person_67' nan nan nan 1.4 22 '2015-07-31' True 58]
 [68 'Person_68' nan nan nan 1.4 15 '2015-08-31' True 95]
 [69 'Person_69' 31.0 76717.0 'Sales' 2.2 38 '2015-09-30' True 50]
 [70 'Person_70' 40.0 80859.0 'Sales' 4.9 38 '2015-10-31' False 94]
 [71 'Person_71' 57.0 56309.0 'IT' 1.7 13 '2015-11-30' True 62]
 [72 'Person_72' 38.0 93734.0 'Marketing' 1.1 30 '2015-12-31' True 53]
 [73 'Person_73' 33.0 82662.0 'HR' 4.1 4 '2016-01-31' True 50]
 [74 'Person_74' 35.0 42688.0 'Sales' 4.2 34 '2016-02-29' True 98]
 [75 'Person_75' 41.0 55342.0 'Sales' 2.4 22 '2016-03-31' False 89]
 [76 'Person_76' 43.0 67157.0 'HR' 2.9 28 '2016-04-30' False 81]
 [77 'Person_77' 42.0 97863.0 'Sales' 3.6 10 '2016-05-31' True 83]
 [78 'Person_78' 58.0 82083.0 'IT' 1.2 17 '2016-06-30' False 77]
 [79 'Person_79' 46.0 95733.0 'Marketing' 4.8 11 '2016-07-31' True 80]
 [80 'Person_80' 32.0 64698.0 'HR' 4.5 8 '2016-08-31' True 57]
 [81 'Person_81' 18.0 52671.0 'Marketing' 2.0 9 '2016-09-30' True 88]
 [82 'Person_82' 42.0 55184.0 'Sales' 1.1 16 '2016-10-31' False 75]
 [83 'Person_83' 24.0 72107.0 'Sales' 4.7 37 '2016-11-30' False 83]
 [84 'Person_84' 26.0 81663.0 'HR' 3.0 6 '2016-12-31' False 52]
 [85 'Person_85' nan nan nan 3.2 12 '2017-01-31' False 99]
 [86 'Person_86' 18.0 79811.0 'Finance' 3.7 39 '2017-02-28' False 61]
 [87 'Person_87' nan nan nan 3.5 8 '2017-03-31' False 50]
 [88 'Person_88' nan nan nan 4.8 26 '2017-04-30' False 93]
 [89 'Person_89' 28.0 64754.0 'Finance' 4.8 1 '2017-05-31' False 54]
 [90 'Person_90' 34.0 41411.0 'IT' 4.5 4 '2017-06-30' False 79]
 [91 'Person_91' 25.0 32911.0 'Finance' 3.5 28 '2017-07-31' True 79]
 [92 'Person_92' 52.0 97270.0 'Finance' 4.2 36 '2017-08-31' False 66]
 [93 'Person_93' 52.0 38680.0 'IT' 3.7 37 '2017-09-30' False 97]
 [94 'Person_94' 50.0 41111.0 'Finance' 3.3 18 '2017-10-31' False 96]
 [95 'Person_95' 22.0 67504.0 'Finance' 1.5 7 '2017-11-30' False 72]
 [96 'Person_96' 59.0 31802.0 'Finance' 4.2 0 '2017-12-31' False 64]
 [97 'Person_97' 56.0 38155.0 'HR' 4.3 21 '2018-01-31' False 86]
 [98 'Person_98' 58.0 69384.0 'HR' 3.5 16 '2018-02-28' False 70]
 [99 'Person_99' 45.0 77254.0 'HR' 4.3 6 '2018-03-31' False 63]
 [100 'Person_100' 24.0 51918.0 'IT' 3.6 24 '2018-04-30' False 51]]
```

```
print("List Equivalent:\n", df.values.tolist())
```

```
List Equivalent:
 [[1, 'Person_1', 56.0, 38392.0, 'Finance', 3.8, 28, '2010-01-31', True, 77], [2, 'Person_2', na
```

```
# 2. Extract row and column number of a particular cell based on a criterion
criterion = df['Age'] == 'Ages'
row_positions = df.index[criterion].tolist()
col_positions = [df.columns.get_loc('Age')]
print("Row Positions:", row_positions)
print("Column Positions:", col_positions)
```

```
Row Positions: []
Column Positions: [2]
```

```
# 3. Rename a specific column
df.rename(columns={'Name': 'Person_Name'}, inplace=True)
```

```
df.head()
```

	ID	Person_Name	Age	Salary	Department	Rating	Experience	Join_Date	Has
0	1	Person_1	56.0	38392.0	Finance	3.8	28	2010-01-31	
1	2	Person_2	NaN	NaN	NaN	3.2	12	2010-02-28	
2	3	Person_3	32.0	82256.0	Marketing	2.2	11	2010-03-31	
3	4	Person_4	25.0	65222.0	Finance	2.7	30	2010-04-30	
4	5	Person_5	38.0	93335.0	HR	2.0	1	2010-05-31	

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 4. Count missing values per column
df.isnull().sum()
```

	0
ID	0
Person_Name	0
Age	10
Salary	10
Department	10
Rating	0
Experience	0
Join_Date	0
Has_Certification	0
Performance_Score	0

dtype: int64

```
# 5. Replace missing values in multiple numeric columns with mean
df['Age'] = df['Age'].fillna(df['Age'].mean())
```

```
df.head()
```

	ID	Person_Name	Age	Salary	Department	Rating	Experience	Join_Date
0	1	Person_1	56.000000	38392.0	Finance	3.8	28	2010-01-31
1	2	Person_2	37.588889	NaN	NaN	3.2	12	2010-02-28
2	3	Person_3	32.000000	82256.0	Marketing	2.2	11	2010-03-31
3	4	Person_4	25.000000	65222.0	Finance	2.7	30	2010-04-30
4	5	Person_5	38.000000	93335.0	HR	2.0	1	2010-05-31

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 6. Replace a missing value using the Imputer class
imputer = SimpleImputer(strategy='mean')
numeric_cols = df.select_dtypes(include=['number']).columns
df[numeric_cols] = imputer.fit_transform(df[numeric_cols])
```

```
df.isnull().sum()
```

	0
ID	0
Person_Name	0
Age	0
Salary	0
Department	10
Rating	0
Experience	0
Join_Date	0
Has_Certification	0
Performance_Score	0

dtype: int64

```
df.head()
```

	ID	Person_Name	Age	Salary	Department	Rating	Experience	Join_
0	1.0	Person_1	56.000000	38392.000000	Finance	3.8	28.0	2010·
1	2.0	Person_2	37.588889	66913.388889	NaN	3.2	12.0	2010·
2	3.0	Person_3	32.000000	82256.000000	Marketing	2.2	11.0	2010·
3	4.0	Person_4	25.000000	65222.000000	Finance	2.7	30.0	2010·
4	5.0	Person_5	38.000000	93335.000000	HR	2.0	1.0	2010·

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 7. Apply function to existing columns with global variables as arguments
global_var = 10
def custom_function(x, multiplier):
    return x * multiplier
df['Updated_Salary'] = df['Salary'].apply(custom_function, args=(global_var,))
df.head()
```

	ID	Person_Name	Age	Salary	Department	Rating	Experience	Join_
0	1.0	Person_1	56.000000	38392.000000	Finance	3.8	28.0	2010·
1	2.0	Person_2	37.588889	66913.388889	NaN	3.2	12.0	2010·
2	3.0	Person_3	32.000000	82256.000000	Marketing	2.2	11.0	2010·
3	4.0	Person_4	25.000000	65222.000000	Finance	2.7	30.0	2010·
4	5.0	Person_5	38.000000	93335.000000	HR	2.0	1.0	2010·

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 8. Change column order while keeping all columns sorted
df = df[sorted(df.columns)]
df.head()
```

	Age	Department	Experience	Has_Certification	ID	Join_Date	Performa
0	56.000000	Finance	28.0	True	1.0	2010-01-31	
1	37.588889	NaN	12.0	False	2.0	2010-02-28	
2	32.000000	Marketing	11.0	False	3.0	2010-03-31	
3	25.000000	Finance	30.0	False	4.0	2010-04-30	
4	38.000000	HR	1.0	False	5.0	2010-05-31	

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 9. Set number of rows & columns displayed
pd.set_option('display.max_rows', 50)
pd.set_option('display.max_columns', 10)
df.head()
```

	Age	Department	Experience	Has_Certification	ID	...	Performance_Sc
0	56.000000	Finance	28.0	True	1.0	...	
1	37.588889	NaN	12.0	False	2.0	...	
2	32.000000	Marketing	11.0	False	3.0	...	
3	25.000000	Finance	30.0	False	4.0	...	
4	38.000000	HR	1.0	False	5.0	...	

5 rows × 11 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 10. Create primary key index by combining columns
df['primary_key'] = df['Person_Name'].astype(str) + '_' + df['Age'].astype(str)
df.set_index('primary_key', inplace=True)
df.head()
```

primary_key	Age	Department	Experience	Has_Certification
Person_1_56.0	56.000000	Finance	28.0	True
Person_2_37.58888888888889	37.588889	NaN	12.0	False
Person_3_32.0	32.000000	Marketing	11.0	False
Person_4_25.0	25.000000	Finance	30.0	False
Person_5_38.0	38.000000	HR	1.0	False

5 rows × 11 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 11. Get row number of nth largest value in a column
n = 5
m=3
row_num = df['Salary'].nlargest(n).index[-1]
print("Row Number of nth largest value:", row_num)

row_num = df['Salary'].nlargest(m).index[-1]
print("Row Number of mth largest value:", row_num)
```

```
Row Number of nth largest value: Person_19_47.0
Row Number of mth largest value: Person_77_42.0
```

```
# 12. Find position of nth largest value greater than a given value
given_value = 50
filtered_df = df[df['Salary'] > given_value]
nth_largest_row = filtered_df['Salary'].nlargest(n).index[-1]
print("Position of nth largest value > given value:", nth_largest_row)
```

```
Position of nth largest value > given value: Person_19_47.0
```

```
# 13. Get last n rows where row sum > 100
n = 5  # Change as needed
filtered_rows = df[df.select_dtypes(include=[np.number]).sum(axis=1) > 100].tail(n)
print(filtered_rows)
```

```
                  Age Department  Experience  Has_Certification     ID  ...  \
primary_key                                                           ...
Person_96_59.0   59.0    Finance         0.0              False   96.0  ...
Person_97_56.0   56.0         HR        21.0              False   97.0  ...
Person_98_58.0   58.0         HR        16.0              False   98.0  ...
Person_99_45.0   45.0         HR         6.0              False   99.0  ...
Person_100_24.0  24.0         IT        24.0              False  100.0  ...

                Performance_Score  Person_Name Rating   Salary  Updated_Salary
primary_key
Person_96_59.0               64.0    Person_96    4.2  31802.0        318020.0
Person_97_56.0               86.0    Person_97    4.3  38155.0        381550.0
Person_98_58.0               70.0    Person_98    3.5  69384.0        693840.0
Person_99_45.0               63.0    Person_99    4.3  77254.0        772540.0
Person_100_24.0              51.0   Person_100    3.6  51918.0        519180.0

[5 rows x 11 columns]
```

```
# 14. Create a column with min/max of each row (only for numeric columns)
numeric_df = df.select_dtypes(include=[np.number])
df['min_max_ratio'] = numeric_df.min(axis=1) / numeric_df.max(axis=1)

df.head()
```

primary_key	Age	Department	Experience	Has_Certification
Person_1_56.0	56.000000	Finance	28.0	True
Person_2_37.58888888888889	37.588889	NaN	12.0	False
Person_3_32.0	32.000000	Marketing	11.0	False
Person_4_25.0	25.000000	Finance	30.0	False
Person_5_38.0	38.000000	HR	1.0	False

5 rows × 12 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Ensure the column exists in your dataset
categorical_col = 'Department'

# Apply Label Encoding
label_encoder = LabelEncoder()
df['encoded_col'] = label_encoder.fit_transform(df[categorical_col])

# Apply One-Hot Encoding
one_hot = pd.get_dummies(df[categorical_col], prefix='category')
df = pd.concat([df, one_hot], axis=1)



# 16. Normalize columns using MinMaxScaler
scaler = MinMaxScaler()
df[['Age', 'Experience']] = scaler.fit_transform(df[['Age', 'Experience']])



df.head()
```

primary_key	Age	Department	Experience	Has_Certification	ID	...
_1_56.0	0.926829	Finance	0.693813	True	-1.714816	...
8888888888889	0.477778	NaN	-0.780424	False	-1.680173	...
_3_32.0	0.341463	Marketing	-0.872564	False	-1.645531	...
_4_25.0	0.170732	Finance	0.878093	False	-1.610888	...
_5_38.0	0.487805	HR	-1.793963	False	-1.576245	...

ıns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 17. Normalize columns using Z-score normalization
scaler = StandardScaler()
df[['Experience', 'ID']] = scaler.fit_transform(df[['Experience', 'ID']])


print(df.columns)
```

```
Index(['Age', 'Department', 'Experience', 'Has_Certification', 'ID',
       'Join_Date', 'Performance_Score', 'Person_Name', 'Rating', 'Salary',
       'Updated_Salary', 'min_max_ratio', 'encoded_col', 'category_Finance',
       'category_HR', 'category_IT', 'category_Marketing', 'category_Sales'],
      dtype='object')
```

```
# 20. Sort values by a specific column in descending order
df.sort_values(by='Experience', ascending=False, inplace=True)

print("Operations completed successfully!")
df.head()
```

```
Operations completed successfully!
```

primary_key	Age	Department	Experience	Has_Certification	ID	...	category_Finance
Person_86_18.0	0.000000	Finance	1.707351	False	1.229818	...	True
Person_32_20.0	0.048780	Finance	1.615211	True	-0.640891	...	True
Person_69_31.0	0.317073	Sales	1.615211	True	0.640891	...	False
Person_70_40.0	0.536585	Sales	1.615211	False	0.675534	...	False
Person_93_52.0	0.829268	IT	1.523072	False	1.472317	...	False

5 rows × 18 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# 18. Import data from URL
df_url = pd.read_html("http://www.fdic.gov/bank/individual/failed/banklist.html")[0]
print(df_url.head())
```

```
                            Bank Name          City         State   Cert  \
0                  Pulaski Savings Bank       Chicago      Illinois  28611
1     The First National Bank of Lindsay       Lindsay      Oklahoma   4134
2  Republic First Bank dba Republic Bank  Philadelphia  Pennsylvania  27332
3                         Citizens Bank      Sac City          Iowa   8758
4              Heartland Tri-State Bank       Elkhart        Kansas  25851

                 Aquiring Institution      Closing Date  Fund  Sort ascending
0                     Millennium Bank  January 17, 2025                 10548
1  First Bank & Trust Co., Duncan, OK  October 18, 2024                 10547
2   Fulton Bank, National Association    April 26, 2024                 10546
3          Iowa Trust & Savings Bank  November 3, 2023                 10545
4             Dream First Bank, N.A.     July 28, 2023                 10544
```

```
print(df_url.columns)
```

```
Index(['Bank Name', 'City', 'State', 'Cert', 'Aquiring Institution',
       'Closing Date', 'Fund  Sort ascending'],
      dtype='object')
```

```
# 18th ki 20 question. Sort values by a specific column in descending order
df_url.sort_values(by='Bank Name', ascending=False, inplace=True)

print("Operations completed successfully!")
```

```
Operations completed successfully!
```

```
print("Unique value count in 'Bank Name':", df_url['Bank Name'].nunique())  # Count of unique values
print(df_url['Bank Name'].value_counts())  # Frequency of each unique value
```

```
Unique value count in 'Bank Name': 10
Bank Name
The First National Bank of Lindsay       1
Silicon Valley Bank                      1
Signature Bank                           1
Republic First Bank dba Republic Bank    1
Pulaski Savings Bank                     1
Heartland Tri-State Bank                 1
First Republic Bank                      1
First City Bank of Florida               1
Citizens Bank                            1
```

```
Almena State Bank                 1
Name: count, dtype: int64
```